

# Distributed-Memory Large Deformation Diffeomorphic 3D Image Registration

Andreas Mang, Amir Gholami, and George Biros  
 The Institute of Computational Engineering and Sciences  
 The University of Texas at Austin, Austin, Texas 78712–1229  
 andreas@ices.utexas.edu; amir@accfft.org; gbiros@acm.org

**Abstract**—We present a parallel distributed-memory algorithm for large deformation diffeomorphic registration of volumetric images that produces large isochoric deformations (locally volume preserving). Image registration is a key technology in medical image analysis. Our algorithm uses a partial differential equation constrained optimal control formulation. Finding the optimal deformation map requires the solution of a highly nonlinear problem that involves pseudo-differential operators, biharmonic operators, and pure advection operators both forward and backward in time. A key issue is the time to solution, which poses the demand for efficient optimization methods as well as an effective utilization of high performance computing resources. To address this problem we use a preconditioned, inexact, Gauss-Newton-Krylov solver. Our algorithm integrates several components: a spectral discretization in space, a semi-Lagrangian formulation in time, analytic adjoints, different regularization functionals (including volume-preserving ones), a spectral preconditioner, a highly optimized distributed Fast Fourier Transform, and a cubic interpolation scheme for the semi-Lagrangian time-stepping. We demonstrate the scalability of our algorithm on images with resolution of up to  $1024^3$  on the “Maverick” and “Stampede” systems at the Texas Advanced Computing Center (TACC). The critical problem in the medical imaging application domain is strong scaling, that is, solving registration problems of a moderate size of  $256^3$ —a typical resolution for medical images. We are able to solve the registration problem for images of this size in less than five seconds on 64 x86 nodes of TACC’s “Maverick” system.

**Index Terms**—Diffeomorphic Image Registration, Optimal Control, Newton-Krylov Methods, Scientific Computing, High Performance Computing.

## I. INTRODUCTION

Deformable registration (also known as image alignment, warping, or matching) refers to methods that find point correspondences between images by comparing image intensities.

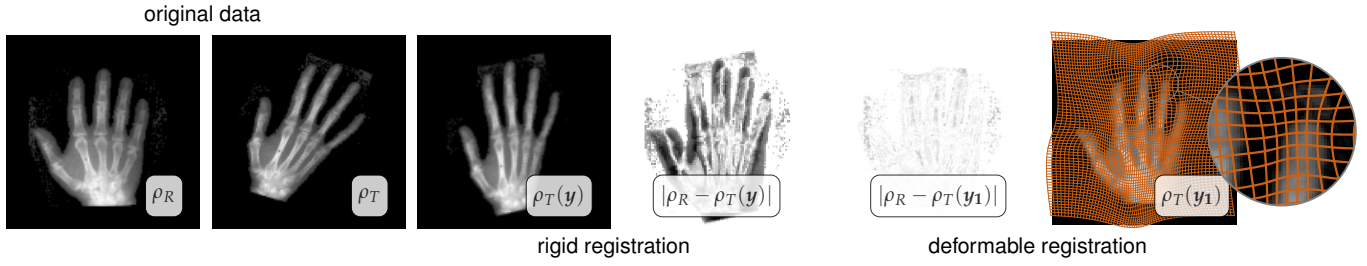
This material is based upon work supported by AFOSR grants FA9550-12-10484 and FA9550-11-10339; by NSF grant CCF-1337393; by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Applied Mathematics program under Award Numbers DE-SC0010518 and DE-SC0009286; by NIH grant 10042242; by DARPA grant W911NF-115-2-0121; and by the Technische Universität München, Institute for Advanced Study, funded by the German Excellence Initiative (and the European Union Seventh Framework Programme under grant agreement 291763). Any opinions, findings, and conclusions or recommendations expressed herein are those of the authors and do not necessarily reflect the views of the AFOSR, the DOE, the NIH, the DARPA, or the NSF. Computing time on the Texas Advanced Computing Centers Stampede system was provided by an allocation from TACC and the NSF.

accepted for publication at SC16; Salt Lake City, Utah, USA; November 2016

We refer to these point correspondences as the *deformation map* (see Figure 1 in §II). An example for a low dimensional image registration problem is affine registration; it creates simple maps consisting of rotations, translations, and scalings [49]. Typically, affine registration is used as an initialization step for *large deformation diffeomorphic registration* (LDDR), which is the problem we are concerned with in the present work. In LDDR, we typically search for a deformation map for which the degrees of freedom are the ambient space times the number of grid points defined in the image space. LDDR is much more flexible than affine registration and thus, in general, more informative in clinical studies [55], [38]. Such high-dimensional transformations can be defined in many different ways [49], [50], [55]. Image registration is an ill-posed inverse problem; it does not have a unique solution. Not all large deformation maps are *admissible* since they can shuffle the points arbitrarily to match intensities. It is crucial to impose constraints on the deformation while allowing for flexibility. The most important constraint is that the map is *diffeomorphic* (see Figure 2 in §II).

Solving an LDDR problem in a rigorous way requires the solution of a non-convex partial differential equation (PDE) constrained optimal control problem [10], [26], [31], [41]. This problem is ill-posed and involves non-linear and ill-conditioned operators. Most state-of-the-art packages circumvent these issues by sacrificing scalability and settling for crude solutions using simple but suboptimal algorithms. In many cases this works sufficiently well, but in several other cases, it does not. There is significant activity in trying to improve the existing algorithms. With regards to performance optimizations, most codes use open multi-processing (OpenMP) or graphics processing unit (GPU) acceleration; there are very few codes that utilize distributed memory parallelism. As a result they are not scalable to the full resolution; to solve problems for large images most codes use subsampling. This is limiting considering the current imaging resolutions. Seven Tesla magnetic resonance imaging (MRI) scanners can reach a resolution of up to 0.5 mm ( $\approx 450^3$  voxels) [44]. Ultra-high resolution computed tomography (CT) captures 0.25 mm resolution ( $\approx 512^3$  voxels) [36].

Beyond the need for strong scaling of image registration



**Fig. 1:** The image registration problem (data taken from [50], [3]). The input (original data) are image intensities  $\rho_R$  and  $\rho_T$ . The output is  $\mathbf{y}$ , the deformation map. Our goal is to find  $\mathbf{y}$  so that  $\rho_T(\mathbf{y})$  (the **deformed**  $\rho_T$ ) is as close as possible to  $\rho_R$  (with respect to some appropriate measure). One way to achieve this is to use rigid registration (i.e., searching for a map that entirely is described by rotations and translations). The result of a rigid registration is shown in the third image from the left; the fourth image shows the difference between the two images  $\rho_R$  and  $\rho_T$  after rigid registration. As one can see, there are still significant differences in the intensity values. If we use a deformable registration instead, we can compute a much more flexible  $\mathbf{y}_1$ , which results in a much smaller misfit  $|\rho_R - \rho_T(\mathbf{y}_1)|$ . The deformation map  $\mathbf{y}_1$  for this method is visualized in the last figure to the right. The grid lines, superimposed on top of  $\rho_T(\mathbf{y}_1)$  were a Cartesian grid before the deformation. We use them to visualize the overall deformation.

algorithms for clinical applications, there is also need for weak scaling for imaging in biology, biophysics, and neuroscience. Animal Micro-CT reaches  $\mathcal{O}(\mu\text{m})$  resolution ( $\approx 2000^3$  voxels) [56], [64]. In small animal neuroimaging, CLARITY [58], a novel optical imaging technique, can deliver sub-micron resolution for the whole brain of the animal, resulting in 10-100 billion-voxel images. To our knowledge, none of the existing schemes for LDDR allow for the registration of such large volumetric images [5], [9].

**Contributions:** The design goals for our 3D LDDR scheme are the following: (1) ability to represent large diffeomorphic deformations; (2) algorithms based on rigorous mathematical foundations; (3) algorithmic optimality with respect to *both* the deformation map resolution and the image resolution; and (4) parallel scalability. Here, we propose an algorithm that achieves these goals and has the following characteristics:

- It is based on optimal control theory. Our formulation allows the control of the registration quality in terms of image correspondence and different quality metrics for the diffeomorphism/deformation map (see §II-A).
- It uses a semi-Lagrangian approach for solving the transport equations that govern the deformation of the image. This approach leads to algorithmic optimality (see §III-B2).
- It uses a spectral discretization in space (see §III-B1). This discretization enables flexibility in the choice of regularization operators for the deformation map. Such flexibility is necessary since different image registration applications have different requirements. It also allows for efficient solvers of saddle-point linear systems.
- It uses optimal algorithms based on an adjoint-based formulation solved via a line-search globalized, inexact, pre-conditioned Gauss-Newton-Krylov scheme (see §III-A).
- It uses distributed-memory parallelism for scalability, employs several performance optimizations specific to our problem, and uses a parallel FFT for elliptic solvers

and differentiations that has been shown to scale to hundreds of thousands of cores (see §III-C). It employs several optimizations for the most expensive part of the computation (cubic interpolation) (see §III-C2). It also supports GPU acceleration (not discussed here).

In addition, we analyze the overall complexity of our method in terms of communication, computation, and storage. The class of deformations we consider here are one of the most challenging since we enable locally volume preserving maps,<sup>1</sup> which find many applications [54], [63], [29], [12]. We present results for synthetic and neurological images and demonstrate the performance of our algorithm (see §IV) for both volume preserving and more generic deformation maps.

**Related work on high performance computing methods for 3D image registration:** A rich literature survey on high performance computing (HPC) in image registration can be found in [53], [18], [35]. General surveys on image registration can be found in [49], [55]. Formulations related to the one discussed in this work are reviewed in [45], [46], [47].

State of the art registration packages that are used in the medical imaging and medical image computing community include ELASTIX [39], ANTS [6], DARTEL [4], and DEMONS [60], [61], [43]. All of these offer some kind of diffeomorphic registration scheme. These packages mostly support OpenMP, but do not use GPUs or Message Passing Interface (MPI) acceleration (exceptions to be discussed below). An important distinction should be made between the *image resolution* (number of voxels) and the *map resolution* (number of degrees of freedom for the map parameterization). In general, the higher the map resolution, the better the registration quality [38], [39] but the harder the optimization problem since it has more degrees of freedom. Most existing codes downsample the map resolution significantly.

<sup>1</sup>In the medical imaging jargon this is referred to as “mass preserving” maps.

The majority of researchers have used GPUs to accelerate the calculations. For example, the solver for the LDDR scheme in [28] uses a preconditioned gradient descent (not Newton) algorithm with a hardware-provided trilinear interpolation on a GPU architecture. It supports the distributed solution of multiple independent LDDR image registration problems (in an embarrassingly parallel way), but does not support distributed memory parallelism for a single LDDR problem.

Two popular packages that exploit GPU acceleration are NIFTYREG [48] and PLASTIMATCH [52]. They use B-spline parameterized low-resolution maps ( $50^3$  coefficients), a tri-linear interpolation scheme, and gradient descent type optimization; NIFTYREG supports soft constraints to penalize volume change.

An MPI version of NIFTYREG for bigger images [33] exists, but the map resolution remains the same ( $50^3$  regular grid for the deformation field). The pioneering works [62], [42] support MPI. Their formulation is based on elastic deformation maps (not LDDR). A GPU-LDDR scheme that supports somewhat high-resolution maps ( $128^3$ ) is [59]. It uses steepest descent (not Newton) and does not support MPI; no timings are reported.

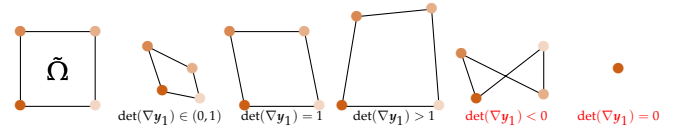
To summarize, existing schemes do not support scalable LDDR algorithms and no scaling studies have been reported.

**Limitations:** In multiframe volume registration (e.g., 4D Cine-MRI) one seeks to register multiple images using a smooth, continuous mapping [13], [9]. Our solver can be used as is, but our diffeomorphic map parameterization is better suited for registering two images. Our parameterization can be extended without any major algorithmic changes but the software engineering would require some work. Another missing piece is a preconditioner that is insensitive to the regularization parameter. There are several techniques for doing so, e.g., grid continuation and multilevel preconditioning [10], [1], [47]. Here we focus on the single-level solver. The single node performance of the interpolation can be improved by more sophisticated blocking, manual vectorization, and possibly prefetching. Similar considerations hold true for the GPU version of the interpolation. This is ongoing work. Another limitation is that we only consider a discretization on Cartesian grids. This is not always the best grid [62]. The structure of our algorithm changes significantly for unstructured grids.

## II. BACKGROUND

Let  $\Omega = [0, 2\pi)^3 \subset \mathbb{R}^3$  be the spatial domain, in which we define functions (images).  $\partial\Omega$  denotes the boundary of  $\Omega$  and  $\mathbf{x}$  a point in  $\Omega$ . Let  $\rho(\mathbf{x}) \in \mathbb{R}$  be a function defined on  $\Omega$ . In imaging  $\rho(\mathbf{x})$  is the **image intensity** at a point  $\mathbf{x}$ ; in optimal control  $\rho(\mathbf{x})$  is the **state field**. In the registration problem, we have a reference image, denoted by  $\rho_R(\mathbf{x})$ , and a template image, denoted by  $\rho_T(\mathbf{x})$ ; the goal is to find a vector valued deformation map, denoted by  $\mathbf{y}(\mathbf{x})$ , that maps a point of the template image  $\rho_T$  to a corresponding point in the reference image  $\rho_R$  [49], [50].

Let  $\mathbf{v}(\mathbf{x})$  be the **velocity field** that generates the map  $\mathbf{y}$ . In our formulation, we introduce a *pseudotime* to denote the



**Fig. 2:** Here we illustrate diffeomorphic and non-diffeomorphic transformations in 2D by considering an infinitesimal area  $\hat{\Omega}$ . We can think of  $\hat{\Omega}$  as a single grid cell (pixel). The first figure on the left depicts the undeformed area. The second figure shows an admissible deformation that shrinks the original area. The third figure depicts an area-preserving deformation (in 3D it would be volume preserving). The fourth figure shows the result of a deformation that expands the area. The fifth illustration is not diffeomorphic since material lines that did not cross each other in the original (leftmost illustration), cross now. The sixth, rightmost, illustration corresponds to a singular deformation in which all the spatial information is lost by shrinking  $\hat{\Omega}$  to a single point. The last two deformations are not useful in image analysis. However, without any constraints on the deformation map, pixels with non-diffeomorphic behavior appear almost always. For this reason appropriate regularization of the problem is necessary.

deformation of the template image at time  $t$ , denoted by  $\rho(\mathbf{x}, t)$ . We define  $\rho(\mathbf{x}, t = 0)$  to be the undeformed template image  $\rho_T$ , and  $\rho(\mathbf{x}, t = 1)$  to be the result of applying the deformation map (which needs to be compared to  $\rho_R(\mathbf{x})$ ).

For the optimal control problem,  $\lambda(\mathbf{x}, t)$  is the **adjoint field**,  $\mathcal{H}$  is the reduced **Hessian operator**,  $\mathbf{g}$  is the **gradient field**, and  $\beta > 0$  is the scalar regularization parameter. We use periodic boundary conditions for all differential operators. For the discretization,  $N_i$  is the **number of grid points** per  $i$ -th dimension;  $N_1 N_2 N_3$  is the total number of unknowns in space;  $n_t$  is the **number of time steps** and  $\delta t$  the **time step size**. We use  $p$  for the number of **MPI tasks**,  $t_s$  for the latency in seconds, and  $t_w$  for the reciprocal of the bandwidth when we do complexity analysis. Boldface lowercase symbols indicate vectors in  $\mathbb{R}^3$ .

### A. Image registration

The image registration problem can be abstractly defined as follows. Given two functions (i.e., images)  $\rho_T(\mathbf{x})$  and  $\rho_R(\mathbf{x})$ , we seek a vector function  $\mathbf{y}_1(\mathbf{x})$  such that the  $L^2$ -distance (i.e., the residual) between  $\rho_T(\mathbf{y}_1(\mathbf{x}))$  and  $\rho_R(\mathbf{x})$  is minimal. We can think of  $\mathbf{y}_1(\mathbf{x})$  as deforming an (infinite) grid of points in the template image  $\rho_T$  so that their intensity after the deformation matches the reference image  $\rho_R$  in the  $L^2$ -norm.<sup>2</sup>

It can be shown that this problem has an infinite number of solutions, most of which are not useful. To resolve this, one needs to impose additional constraints, such as smoothness (i.e.,  $\nabla \mathbf{y}_1(\mathbf{x})$  exists), although this alone may not be sufficient

<sup>2</sup>Other types of *distance measures* can be used (see, e.g., [49], [50], [55]). There are no significant changes in our formulation or algorithm if we would consider other, popular distance measures.



to ensure that the map is plausible. Note, that we require  $\det(\nabla \mathbf{y}_1(\mathbf{x})) > 0$  for all  $\mathbf{x} \in \Omega$  to guarantee that  $\mathbf{y}_1(\mathbf{x})$  is a *diffeomorphism* (for an illustration, see 2). In velocity-based LDDR it can be shown that such a diffeomorphic map  $\mathbf{y}_1$  exists, if the generating **velocity field**  $\mathbf{v}(\mathbf{x}, t)$  is adequately smooth [9], [15], [13]; a typical requirement is that  $\mathbf{v}$  is an  $H^2$ -function [9]. The deformation map  $\mathbf{y}_1$  can be computed from  $\mathbf{v}$  by solving

$$\partial_t \mathbf{y}(\mathbf{x}, t) + \mathbf{v}(\mathbf{x}, t) \cdot \nabla \mathbf{y}(\mathbf{x}, t) = 0, \quad \mathbf{y}(\mathbf{x}, 0) = \mathbf{x}, \quad (1)$$

where  $\mathbf{y}_1(\mathbf{x}) = \mathbf{y}(\mathbf{x}, 1)$ . It can be shown that if the velocity is *incompressible* (i.e.,  $\text{div } \mathbf{v} = 0$  for every  $\mathbf{x}$  in  $\Omega$ ), then  $\det \nabla \mathbf{y}(\mathbf{x}, t) = 1$  and the diffeomorphism is referred to as *volume preserving* [27]. Here, we consider both the general and the incompressible velocity cases. The latter case is more challenging. We only consider stationary velocity fields, that is,  $\mathbf{v}(\mathbf{x}, t) = \mathbf{v}(\mathbf{x})$ .<sup>3</sup>

In the following we drop the dependence of the functions on the spatial position  $\mathbf{x}$  for notational convenience.

### B. Formulation

The solution to the image registration problem can be found by solving the following PDE-constrained optimization problem [30], [13], [45]:

$$\min_{\mathbf{v}} \mathcal{J}[\mathbf{v}] = \frac{1}{2} \|\rho_1 - \rho_R\|_{L^2(\Omega)}^2 + \frac{\beta}{2} \|\Delta \mathbf{v}\|_{L^2(\Omega)}^2 \quad (2a)$$

subject to

$$\partial_t \rho(t) + \mathbf{v} \cdot \nabla \rho(t) = 0 \quad \text{in } \Omega \times (0, 1], \quad (2b)$$

$$\rho(0) = \rho_T \quad \text{in } \Omega, \quad (2c)$$

$$\text{div } \mathbf{v} = 0 \quad \text{in } \Omega. \quad (2d)$$

The second term of  $\mathcal{J}$  enforces smoothness for  $\mathbf{v}$  and  $\beta > 0$  is the *regularization parameter*. In this formulation,  $\rho_1(\mathbf{x}) = \rho(\mathbf{x}, 1) \equiv \rho_T(\mathbf{y}_1)$ , where  $\mathbf{y}_1$  is the solution of (1) at  $t = 1$ . The constraint (2b) defines an implicit function between  $\rho_1$  and  $\mathbf{v}$ ; given  $\mathbf{v}$  we solve (2b) for  $\rho$ .

a) *Computing the gradient of  $\mathcal{J}$* : Given  $\mathbf{v}$ , we need several steps to compute the gradient  $\mathbf{g} = \partial_{\mathbf{v}} \mathcal{J}$ . First we compute  $\rho(1)$  by solving (2b) (with initial condition defined by (2c)). Then we compute the adjoint function  $\lambda(t)$  by solving the **backward-in-time adjoint equation** [34], [10]

$$\begin{aligned} -\partial_t \lambda(t) - \text{div}(\mathbf{v} \lambda(t)) &= 0 & \text{in } \Omega \times [0, 1), \\ \lambda(1) &= \rho_R - \rho(1) & \text{in } \Omega. \end{aligned} \quad (3)$$

Once we have the state and adjoint fields, we can evaluate the **gradient** given by

$$\mathbf{g}(\mathbf{v}) := \beta \Delta^2 \mathbf{v} + (I - \nabla \Delta^{-1} \text{div}) \int_0^1 \lambda(t) \nabla \rho(t) dt. \quad (4)$$

<sup>3</sup>It can be shown that the space of possible diffeomorphisms generated by time-varying velocities is strictly larger than the space generated by stationary velocities. This does not have practical implications when we register two images (see, e.g., [45]). However, it is restrictive if we want register a sequence of images, like in optical flow problems.

The operator  $\mathcal{P} = I - \nabla \Delta^{-1} \text{div}$ , also known as the *Leray operator* [57], eliminates the incompressibility constraint for  $\mathbf{v}$ . Furthermore, we define the vector field

$$\mathbf{b}(\mathbf{x}) := \int_0^1 \lambda(\mathbf{x}, t) \nabla \rho(\mathbf{x}, t) dt,$$

so that  $\mathbf{g}(\mathbf{v}) = \beta \Delta^2 \mathbf{v} + \mathcal{P} \mathbf{b}$ . The gradient  $\mathbf{g}$  is a nonlinear elliptic integro-differential operator where the state (forward in time) and adjoint (backward in time) transport PDEs are “hidden” in  $\mathbf{b}$ .

Evaluating the gradient requires solving two transport equations, inverting the Laplacian and applying gradient, divergence, and biharmonic operators. (If we do not wish to compute a volume preserving map we can drop (i.e., not enforce) the incompressibility constraint. Then, in the gradient calculation, we only need to replace the  $\mathcal{P}$  operator with an identity operator.) **The first-order optimality condition** for (2) requires that  $\mathbf{g}(\mathbf{v}) = \mathbf{0}$ . Most registration packages use steepest descent (first order) methods to find an optimal point (minimizer) [6], [9], [13]. However, steepest descent methods only have a linear convergence rate. Using Newton methods, which provide a much better convergence rate, is considered to be prohibitive, especially for LDDR for two main reasons. First, it is cumbersome to derive the equations for the second order optimality conditions. Second, a naive implementation of Newton methods can be very costly if not done carefully.

#### b) The Newton and Gauss-Newton Hessian operators $\mathcal{H}$ :

To solve  $\mathbf{g}(\mathbf{v}) = \mathbf{0}$  for  $\mathbf{v}$  we use an Armijo line-search globalized Newton method [51]. The key operation is the action of the **Hessian**  $\mathcal{H}(\mathbf{v})$  on a vector field  $\tilde{\mathbf{v}}$ , which is commonly referred to as the **Hessian matvec**. This matvec is computed by performing the following steps: First of all, we need to solve (2b) and (3) to compute the state and adjoint variables  $\rho$  and  $\lambda$ , respectively. After computing these fields, we need to solve (5a) for the *incremental state* variable  $\tilde{\rho}$  and (5c) for the *incremental adjoint* variable  $\tilde{\lambda}$ , accumulating them in time to compute  $\tilde{\mathbf{b}}$  and finally evaluating (5e).

$$\partial_t \tilde{\rho}(t) + \mathbf{v} \cdot \nabla \tilde{\rho}(t) + \tilde{\mathbf{v}} \cdot \nabla \rho(t) = 0 \quad \text{in } \Omega \times (0, 1], \quad (5a)$$

$$\tilde{\rho}(0) = 0 \quad \text{in } \Omega, \quad (5b)$$

$$-\partial_t \tilde{\lambda}(t) - \text{div}(\tilde{\lambda}(t) \mathbf{v} + \lambda(t) \tilde{\mathbf{v}}) = 0 \quad \text{in } \Omega \times [0, 1), \quad (5c)$$

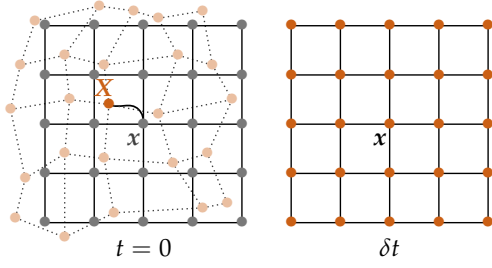
$$\tilde{\lambda}(1) + \tilde{\rho}(1) = 0 \quad \text{in } \Omega, \quad (5d)$$

$$\mathcal{H}(\mathbf{v}) \tilde{\mathbf{v}} := \beta \Delta^2 \tilde{\mathbf{v}} + \mathcal{P} \tilde{\mathbf{b}} \quad \text{in } \Omega, \quad (5e)$$

where

$$\tilde{\mathbf{b}} = \int_0^1 \tilde{\lambda}(t) \nabla \rho(t) + \lambda(t) \nabla \tilde{\rho}(t) dt.$$

Notice that (5a) and (5c) require storing  $\rho(t)$ ,  $\tilde{\rho}(t)$ , and  $\lambda(t)$  for all  $t$ . Also notice that certain terms in (5) drop if we enforce  $\text{div } \mathbf{v} = 0$ . The **Newton step**,  $\tilde{\mathbf{v}}$ , is obtained by **solving the linear system**  $\mathcal{H}(\mathbf{v}) \tilde{\mathbf{v}} = -\mathbf{g}(\mathbf{v})$ . For a **Gauss-Newton** Hessian, we drop the two terms that involve  $\lambda(t)$ , i.e., the last term in (5c) and the second term in  $\tilde{\mathbf{b}}$ .



**Fig. 3:** Illustration of the semi-Lagrangian scheme (figure modified from [47]). The square domain corresponds to a grid block assigned to a single MPI task. Point  $x$  is a regular grid point and  $X$  is the material point at  $t = 0$  that landed at  $x$  at time  $\delta t$ . The semi-Lagrangian algorithm requires interpolation of  $v, \rho, \tilde{\rho}, \lambda$ , and  $\tilde{\lambda}$  at these off-grid points; as illustrated here, these points can lie in other MPI tasks. We communicate these points, interpolate their values, and then communicate them back.

### III. METHODS

Given two images  $\rho_R$  and  $\rho_T$  our goal is to find  $y_1$ . We use an optimize-then-discretize approach for (2). Our scheme can be summarized as follows:

- We solve (2) for  $v$ .
- Once we have  $v$ , we use (1) to compute  $y_1$ .
- To find  $v$  we solve  $g(v) = 0$  (where  $g(v)$  is given by (4)) using a preconditioned Newton-Krylov method.
- In space we use Fourier expansions (regular grids with periodic boundary conditions) and in time we use a semi-Lagrangian scheme.
- We use data parallelism in space (the regular grid).
- All spatial differential operators (and their inverses if needed) are computed spectrally using our parallel FFT.
- All spatial algebraic operators are done in parallel.

We provide more details for our algorithm in the following subsections.

#### A. Newton-Krylov solver and preconditioning

For the optimization we use a Newton method globalized with an Armijo line-search. We use a preconditioned Conjugate-Gradient (PCG) method to compute the Newton step. The linear solves using PCG are done inexactly using a tolerance that depends on the relative norm of the gradient [17]. The preconditioner is the inverse of the biharmonic operator ( $\Delta^{-2}$ ) and can be applied in nearly linear time using FFTs (with a logarithmic factor). This preconditioner delivers mesh-independence—but not  $\beta$ -independence (see §IV). Since the problem is highly nonlinear we use parameter continuation on  $\beta$ . The target value for  $\beta$  is application dependent and, in our algorithm, determined by various metrics defined on  $\nabla y_1$  [45]. We use the TAO module from the PETSc library [8], [7] for numerical optimization, which supports user-defined, matrix free PCG. TAO provides interfaces that allow one to control two main parameters in the Newton-Krylov solver: (i) the accuracy of the solution of the linear

system to compute the Newton step (the relative tolerance of the PCG method used to solve the Hessian equation); and (ii) the nonlinear termination criteria. We provide the algorithms to determine these parameters and, given  $v$  and  $\tilde{v}$ , efficient routines for the function evaluation  $\mathcal{J}(v)$ , the gradient evaluation  $g(v)$ , the Hessian matvec  $\mathcal{H}(v)\tilde{v}$  and the action of the preconditioner  $\Delta^{-2}\tilde{v}$ .

#### B. Discretization

We use Cartesian grids to approximate spatial functions and spatial differential operators. We use an explicit Runge-Kutta second-order semi-Lagrangian method to discretize in time.

1) *Space*: We use a spectral projection scheme for all spatial operations on a regular grid defined on  $\Omega$  with periodic boundary conditions. For simplicity, we consider the isotropic case, in which the grid spacing is the same in all directions; that is, the **number of points per direction** is given by  $N_1 = N_2 = N_3 = N$ . Our actual implementation does not require this (see §IV for an example). We approximate

$$\rho(x) = \sum_{\mathbf{k}} \hat{\rho}_{\mathbf{k}} \exp(-\mathbf{k} \cdot \mathbf{x}),$$

where  $\mathbf{k} = (k_1, k_2, k_3) \in \mathbb{N}^3$  is a multi-index with  $-N/2 + 1 \leq k_j \leq N/2, j = 1, 2, 3$ . The corresponding **regular grid points** are given by  $\mathbf{x}_i = 2\pi\mathbf{i}/N$ , where  $\mathbf{i} = (i_1, i_2, i_3) \in \mathbb{N}^3$  and  $0 \leq i_j \leq N - 1, j = 1, 2, 3$ .

We refer to  $\{\hat{\rho}_{\mathbf{k}}\}$  as the **spectral coefficients** of  $\rho$ . Mappings between  $\{\rho_i\}$  and  $\{\hat{\rho}_{\mathbf{k}}\}$  are done using the forward and inverse **Fast Fourier Transform (FFT)**. We use a similar spectral discretization for  $\lambda$  and for each component of the velocity field  $v$ . All derivatives are performed by first taking the FFT and then filtering the spectral coefficients appropriately. In general, the input images  $\rho_R$  and  $\rho_T$  may not be periodic functions. In that case a spectral approximation will create excessively high aliasing errors. To address this, we use zero-padding for  $\rho_R$  and  $\rho_T$ . Also, in general, images will have discontinuities and thus are not differentiable, creating similar aliasing problems. So, before applying our algorithm, we smooth them spectrally with a Gaussian filter whose bandwidth is  $2\pi/N$  (the grid size). Notice that our spectral representation with periodic boundary conditions allows us to apply all the different spatial operators—including  $\Delta^{-1}$  and  $\Delta^{-2}$ —in a stable, accurate, and extremely efficient manner. As a result, the main cost of the computation will be solving the transport equations, not applying and inverting elliptic differential operators.

2) *Time*: We choose a **semi-Lagrangian** method since it is unconditionally stable [19] and allows us to take a small number of time steps  $n_t \in \mathbb{N}$ . This is critical since we store several space-time fields. For example, when solving (5a) for  $\tilde{\rho}(t)$ , we need  $\rho(t)$  for all  $t$ . For large  $n_t$  the storage requirements become excessive and more sophisticated checkpointing schemes [2] are required—which are more expensive. If we were using a Courant-Friedrichs-Lewy (CFL) restricted<sup>4</sup> scheme for solving the transport equations, storing the time

<sup>4</sup>The CFL condition defines an upper bound on the time step size to ensure a stable solution of stiff, time-dependent PDEs [40].

history would have been impossible, since we had to store hundreds of time steps (see, e.g., [45], [46], [47]).

To explain the semi-Lagrangian method we reintroduce the notational dependence in space. We consider the general transport equation for a scalar field  $\nu(\mathbf{x}, t)$  (with a stationary velocity)

$$\partial_t \nu(\mathbf{x}, t) + \mathbf{v}(\mathbf{x}) \cdot \nabla \nu(\mathbf{x}, t) = f(\nu(\mathbf{x}, t), \mathbf{x}).$$

For example, for (5c) we have  $\nu = \tilde{\lambda}$  and  $f = \tilde{\lambda} \operatorname{div} \mathbf{v} + \operatorname{div}(\tilde{\lambda} \tilde{\mathbf{v}})$ . Then, for each  $\mathbf{x}$ , to compute  $\nu(\mathbf{x}, \delta t)$  given  $\nu(\mathbf{x}, 0)$ , we first compute a new point  $\mathbf{X}$ , **the semi-Lagrangian point**, using the scheme below:

$$\begin{aligned} \mathbf{X}_* &= \mathbf{x} - \delta t \mathbf{v}(\mathbf{x}); \\ \mathbf{X} &= \mathbf{x} - \frac{\delta t}{2} (\mathbf{v}(\mathbf{x}) + \mathbf{v}(\mathbf{X}_*)), \end{aligned} \quad (6)$$

and then we set

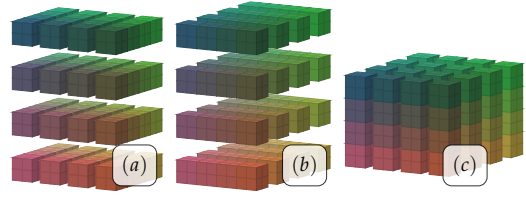
$$\begin{aligned} \nu_0(\mathbf{X}) &= \nu(\mathbf{X}, 0); \\ f_0(\mathbf{X}) &= f(\nu_0(\mathbf{X}), \mathbf{X}); \\ \nu_*(\mathbf{x}) &= \nu_0(\mathbf{X}) + \delta t f_0(\mathbf{X}); \\ f_*(\mathbf{x}) &= f(\nu_*(\mathbf{x}), \mathbf{x}); \\ \nu(\mathbf{x}, \delta t) &= \nu_0(\mathbf{X}) + \frac{\delta t}{2} (f_0(\mathbf{X}) + f_*(\mathbf{x})). \end{aligned} \quad (7)$$

This scheme is fully explicit and *unconditionally stable*. Recall that  $\mathbf{x}$  is a regular grid point and thus  $\nu(\mathbf{x}, 0)$  and  $\mathbf{v}(\mathbf{x})$  are known.  $\mathbf{X}_*$  and  $\mathbf{X}$  are not regular grid points. Computing  $\mathbf{v}$  and  $\nu$  at these off-grid locations requires multiple interpolations: three interpolations for  $\mathbf{v}(\mathbf{X}_*)$ , interpolations for the  $f$  terms that depend on the semi-Lagrangian point  $\mathbf{X}$ , and finally one interpolation for  $\nu(\mathbf{X}, 0)$ . If  $f$  depends on derivatives of  $\nu$ , we first differentiate on the regular grid and then we interpolate the derivatives. The same scheme is used for the adjoint equations by changing the time variable from  $t$  to  $\tau = 1 - t$ , so that  $-\partial_t \lambda(t) = \partial_\tau \lambda(\tau)$  and  $\lambda(t = 1) = \lambda(\tau = 0)$ . Note that the interpolation cannot be done using a FFT, since the interpolation points can be spaced irregularly between grid points.

Cubic interpolation is typically preferred, compared to linear interpolation, because the interpolation errors will be accumulated throughout the time stepping without a time-step factor [11]. We use a **tricubic interpolation scheme**, which we discuss in §III-C2.

### C. Parallel algorithms

The main computational kernels are the 3D FFTs to compute derivatives, elliptic operators and their inverses, the interpolation to off-grid points needed for the semi-Lagrangian time stepping, the Krylov solver (PCG) for the Hessian, and the Newton solver. The 3D FFT has well-known algorithmic complexity. The interpolation on semi-Lagrangian points is the most expensive parts of the computation, despite being local. As it turns out, about 60% of the overall time for the image registration problem is spent on interpolation. The Krylov and Newton solvers are sequential across iterations whereas all the function, gradient, and Hessian evaluations are done using



**Fig. 4:** Here we explain the data partitioning, which is based on the pencil decomposition for 3D FFTs [25]. The colors indicate the data partitioning; each color corresponds to the data assigned to an MPI task. Subfigure (a) represents the input distribution of the (volumetric) image. After an FFT in the first coordinate, in (b) we do the FFT in the second coordinate. This requires  $\sqrt{p}$  concurrent alltoall between groups of  $\sqrt{p}$  MPI tasks. This process is repeated for the third direction in (c) and has the same communication costs as (b) (image modified from [23]).

data parallelism. Below we give more details on the FFT, the interpolation, and how we put everything together.

1) **Partitioning and FFT:** Let  $N_i$ ,  $i = 1, 2, 3$ , be the number of grid points in the  $i^{\text{th}}$  dimension. Also assume we have  $p = p_1 p_2$  MPI tasks. We partition the data using the **pencil decomposition** of 3D FFT (see Figure 4). Each MPI task gets  $(N_1/p_1)(N_2/p_2)N_3$  grid points. There is no partitioning in time and all the time steps are stored in memory. The scalability of the 3D FFT has been well studied [16], [25]. The 3D FFT requires  $\mathcal{O}(\frac{7.5N^3}{p} \log N)$  computations and  $\mathcal{O}(t_s \sqrt{p} + t_w \frac{3N^3}{p})$  communications. We use the open-source package AccFFT [24], [22] that supports both GPU and CPU FFTs and is based on the 1D FFTs implemented in the FFTW package [21]. Our code features optimizations for the  $\nabla$  and  $\operatorname{div}$  operators that allow us to avoid multiple 3D FFTs. For example,  $\nabla \rho = (\partial_1 \rho, \partial_2 \rho, \partial_3 \rho)$  requires  $N_2 N_3$  1D FFTs across the first coordinate, diagonal scaling, and then the same number of inverse FFTs again across the first dimension. A similar process is required for the other components but they require collective all-to-all communications for rearranging the data. The remaining operators  $\mathcal{P}$ ,  $\Delta^2$ , and  $\Delta^{-2}$  are diagonal and require standard 3D FFTs.

2) **Interpolation:** For every grid point  $\mathbf{x}_i$  we have to find points  $\mathbf{X}_i$  required in (7) using (6). In the distributed case, every processor interpolates all the points that fall into the region defined by its pencil (that would be subfigure (a) in Figure 4). This is essentially an alltoallv operation. We refer to this step as “scatter” phase. Note that the points need to be constructed only when the velocity field changes. In a Newton iteration for a given  $\mathbf{v}$  we have to compute these points only once for  $\mathbf{v}$  (forward transport) and for  $-\mathbf{v}$  (adjoint equations); that is the scatter phase needs to be done once per field per Newton iteration. This results in speedups due to savings in communication and computation. After the scatter phase is completed, each process has to perform a cubic interpolation on the points that it owns locally as well as the points it received from the other processors. After this step is

done, an `alltoallv` operation is necessary to send/recv all the interpolation results. This needs to be done once per time step.

The computation is organized as follows. For every forward or adjoint solve, we invoke an *interpolation planner*, which performs the scatter phase and stores the semi-Lagrangian points and creates the communication plans for the transport equation. Then the actual transport (7), which involves multiple interpolations at every time step, is performed. For divergence-free  $\mathbf{v}$ , the computation of (2b) and (3) involves only interpolations. For (5a) and (5c) it also involves differential operators for the gradient and divergence operators that appear on the right-hand side.

Note that it is possible for an interpolation point to fall in-between the locally owned domains of the processes. This is because the local domain of each process is disjoint from others. For this reason, every processor maintains a layer of *ghost points*, regular grid points that belong to other processors. The values of  $\nu$  at these points must be synchronized before interpolation takes place. Notice that for every point we have to bring in 64 scalar values and perform roughly  $10 \times 64$  floating point operations. The constant is related to the 64 coefficients ( $4^3$ ) required to build and evaluate the tricubic interpolant times five flops per coefficient. Therefore, the computation to memory traffic ratio will be  $\mathcal{O}(1)$ —the computation is memory bound. Blocking, prefetching, and vectorization can be used to improve the performance.

---

**Algorithm 1** Parallel tricubic interpolation.

---

**Input:**  $\{\mathbf{X}_{i'}\} \in \Omega_r$ ,  $\text{owner}(\mathbf{X}_{i'})$ ,  $\{\mathbf{x}_i\} \in \Omega_r$ ,  $\text{worker}(\mathbf{x}_i)$ ,  $\nu(\mathbf{x}_i)$ , MPI task  $r$

**Output:**  $\nu(\mathbf{X}_i)$

- 1: Communicate  $\nu$  values for ghost points.
  - 2: Send/recv  $\mathbf{X}_{i'}$  from  $r$  to/from  $\text{owner}(\mathbf{X}_{i'})$ .
  - 3: Locally interpolate to compute  $\nu(\mathbf{X}_i)$ .
  - 4: Send/recv  $\nu(\mathbf{X}_i)$  to/from  $\text{worker}(\mathbf{x}_i)$  to  $r$ .
- 

The execution flow of the interpolation algorithm is explained in Algorithm 1, for an MPI task  $r$ . Let  $\nu(\mathbf{x}_i)$  be the value of the scalar field  $\nu$  at a regular grid point  $\mathbf{x}_i$ . Also, let  $\mathbf{X}_{i'}$  be the corresponding semi-Lagrangian points for each  $i'$  computed by (6). Let  $\Omega_j$  be the spatial domain assigned to the MPI task  $j$ . As shown in figure 3,  $\mathbf{X}_{i'} \in \Omega_r$  does not imply that  $\mathbf{x}_{i'} \in \Omega_r$  and vice versa. For an off-grid point  $\mathbf{X}_{i'}$ ,  $\text{owner}(\mathbf{X}_{i'})$  computes the MPI task that owns that point. That is,  $\mathbf{X}_{i'} \in \Omega_r$  but  $\mathbf{x}_{i'} \in \Omega_{\text{owner}(\mathbf{X}_{i'})}$ . Similarly,  $\mathbf{x}_i \in \Omega$  but  $\mathbf{X}_i \in \Omega_{\text{worker}(\mathbf{x}_i)}$ . Of course for most points both the owner and worker domains will be identical to  $\Omega_r$ . In line 1 every task sends values of  $\nu$  that it owns to its four neighbors (recall that we use a pencil decomposition) with a communication cost of  $4(t_w N^2/p + t_s)$  (the four corner neighbors can be combined with the messages of the edge neighbors, but appropriate ordering of the messages). In line 2, for  $\mathbf{X}_{i'}$  whose corresponding regular grid point  $\mathbf{x}_i$  belongs to a different MPI task,  $r$  sends  $\nu(\mathbf{X}_{i'})$  to that task.

In line 3 the interpolation takes place. This phase requires roughly  $\mathcal{O}(600N^3/p)$  floating point operations. This is followed by an `alltoall` communication in line 4 to send/recv interpolation results. Our GPU implementation follows a similar strategy.

3) *Algorithm for incremental state equation:* We summarize the steps needed to solve the incremental forward problem (5a) for one time step in algorithm 2 to illustrate the communication and computation pattern.

---

**Algorithm 2** One time step of the solution of the incremental forward problem.

---

**Input:**  $\mathbf{v}(\mathbf{x}_i)$ ,  $\tilde{\mathbf{v}}(\mathbf{x}_i)$ ,  $\rho(\mathbf{x}_i, 0)$ ,  $\rho(\mathbf{x}_i, \delta t)$ ,  $\tilde{\rho}(\mathbf{x}_i, 0)$ ,  $\mathbf{X}_i$

**Output:**  $\tilde{\rho}(\mathbf{x}_i, \delta t)$

- 1:  $\rho_0(\mathbf{X}_i) = \rho(\mathbf{X}_i, 0)$  using Algorithm 1.
  - 2:  $\nabla \rho(\mathbf{x}_i, 0)$  using FFT.
  - 3:  $f_0(\mathbf{x}_i) = -\tilde{\mathbf{v}}(\mathbf{x}_i) \cdot \nabla \rho(\mathbf{x}_i, 0)$ .
  - 4:  $f_0(\mathbf{X}_i)$  using  $f_0(\mathbf{x}_i)$  and Algorithm 1.
  - 5:  $\rho_*(\mathbf{x}_i) = \rho_0(\mathbf{X}_i) + \delta t f_0(\mathbf{X}_i)$ .
  - 6:  $\nabla \rho_*(\mathbf{x}_i)$  using FFT.
  - 7:  $f_*(\mathbf{x}_i) = -\tilde{\mathbf{v}}(\mathbf{x}_i) \cdot \nabla \rho_*(\mathbf{x}_i)$ .
  - 8:  $\tilde{\rho}(\mathbf{x}_i, \delta t) = \rho_0(\mathbf{X}_i) + \frac{\delta t}{2} (f_0(\mathbf{X}_i) + f_*(\mathbf{x}_i))$ .
- 

This calculation requires four interpolation steps using Algorithm 1, one for the scalar interpolation in line 1 and three for the vector interpolation in line 4. It also requires four FFTs: two for line 2 (it is two because we need to go the spectral domain, differentiate, and then back to the spatial domain) and two for line 6. The other parts are triple “for-loops” over all the grid points  $i$  in  $\Omega_r$ . The FFTs require global synchronizations. The **total cost of the incremental adjoint solve** is four 3D FFTs and two interpolations. The incremental adjoint requires the same computations (for divergence-free velocity fields).

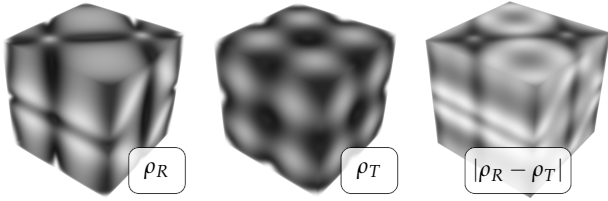
4) *Complexity of Hessian matvec and overall algorithm:* Every **Hessian matvec** requires  $n_t$  forward and adjoint solves or  $8n_t$  FFTs and  $4n_t$  interpolations. The remaining operations of applying the regularization and the preconditioner are negligible since they include just 2 FFTs each. The gradient is also cheaper since (2b) and (3) are simpler than the ones in (5). Regarding **memory**, every task needs to store  $2n_t N^3/p + 5N^3/p$  values for the incremental adjoint and state variables. Therefore, accounting the complexities for the FFT and interpolation we obtain,

$$T_{\text{flop}} \approx n_t \left( 8 \frac{7.5N^3}{p} \log N + 4 \frac{600N^3}{p} \right)$$

$$T_{\text{mpi}} \approx 8n_t \left( 3t_s \sqrt{p} + t_w \frac{3N^3}{p} \right) + 4n_t \left( t_s + t_w \frac{N^2}{p} \right)$$

This estimate assumes that the semi-Lagrangian points are uniformly distributed across processors, however, this is not guaranteed and depends on the velocity field and the CFL number. In practice the interpolation is the predominant cost of the calculation, at least for the problem sizes we have tested. For fixed  $\beta$  the number of Newton iterations are independent of the mesh size, the inversion of highly ill-conditioned operators is done in linear time.





**Fig. 5:** 3D visualization of a synthetic registration problem (volume rendering). From left to right: (i) reference image  $\rho_R$ , (ii) template image  $\rho_T$ , and (iii) initial (before registration) residual differences between  $\rho_R$  and  $\rho_T$ . The reference image  $\rho_R$  is generated from  $\rho_T$  by solving the forward problem with a known velocity  $\mathbf{v}^*$  (details can be found in the text). Dark areas indicate large residual differences and white areas zero residual differences.

## IV. RESULTS

### A. Experimental setup

In this section, we give details on the experimental setup we used to test our solver.

1) *Images:* We use one real-world and one synthetic image to test our algorithm. For the synthetic case we construct the template image as follows:  $\rho_T(\mathbf{x}) = (\sin^2(x_1) + \sin^2(x_2) + \sin^2(x_3))/3$ ; the velocity is given by  $\mathbf{v}^*(\mathbf{x}) = (\cos(x_1)\sin(x_2), \cos(x_2)\sin(x_1), \cos(x_1)\sin(x_3))^T$ ; the reference image  $\rho_R$  is the solution of (2b) with the exact velocity  $\mathbf{v}^*$  (see Figure 5 for an illustration of this problem).<sup>5</sup> We use a synthetic case to perform the scaling studies, since medical images come with a fixed resolution/grid size. To test our scheme on real medical images, we use two 3D MRI brain images of different individuals (“multi-subject registration problem”; grid size:  $256 \times 300 \times 256$ ). This data is from the *Non-rigid Registration Evaluation Project (NIREP)* [14].<sup>6</sup> (see Figure 6 for an illustration).

2) *Implementation and Hardware:* Our code is implemented in C++ and uses MPI and the OpenMP library for multithreading. The code is compiled with the Intel C++ compiler using the `-O3` flag. Although we have GPU implementations both for the FFT and the interpolation, we have not used accelerators in the results we report here. We carry out runtime experiments on the TACC’s “Maverick” system. Each compute node contains dual, ten-core Intel Xeon E5-2680 v2 (Ivy Bridge) processors running at 2.8GHz with 12.8GB/core of memory. Each node also has an NVIDIA Tesla K40 GPU accelerator. We also report large-scale runs on TACC’s “Stampede” system (two eight-core Xeon E5-2680 v1 (Sandy Bridge) processors with 32GB host memory per node). As mentioned before, we use PETSc’s TAO for the nonlinear optimization, vector operations of PETSc for vector

<sup>5</sup>For the incompressible case we use a similar but divergence free velocity field  $\mathbf{v}^*$ .

<sup>6</sup>The data is available at <http://nirep.org>; the interested reader is referred to [14] for more details. We consider the first two datasets na01 and na02 from this repository.

linear operations, and AccFFT for the Fourier transforms. The basic interface to TAO is the functional, gradient, Hessian matvec, and preconditioner, as well as routines to select the tolerances for the nonlinear solver and for the Newton steps.

3) *Parameters:* The regularization parameter  $\beta$  is set to  $1\text{E}-2$  for the scalability runs (for both, the synthetic and the real-world brain example). The number of time steps  $n_t$  controls the accuracy and should be related to the CFL number. For simplicity and to be able to compare different cases, we have kept it fixed to  $n_t = 4$ . The gradient tolerance is  $\mathbf{g}_{\text{tol}} = 1\text{E}-2$  unless otherwise stated. We use an inexact Newton method with quadratic forcing. We do not report continuation results; all the runs are done for a single (experimentally determined) value of  $\beta$ . We report (i) wall-clock times, (ii) communication times, as well as (iii) the time to solution for our method with respect to different registration problems and parameter settings. Since the problem is non-convex and we are not interested in high-accuracy solutions, we opt for a Gauss-Newton approximation.<sup>7</sup>

### B. Scalability using synthetic images

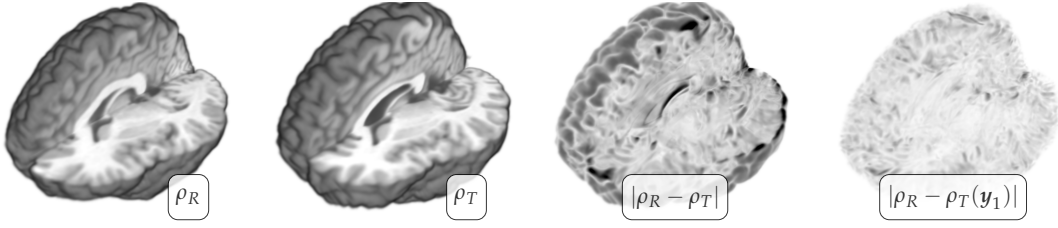
We use a fixed set of parameters, which we experimentally determined to yield a good balance between computational complexity and computational performance. We illustrate the registration problem in Figure 5. We report results for different grid resolutions ( $N_i \in \{64, 128, 256, 512, 1024\}$ ), and different numbers of cores and MPI task configurations ( $p \in \{1, 2, 4, 8, 16, 64, 256\}$ ). The results are reported in Table I (“Maverick” runs) and Table II (large scale “Stampede” runs). First, we interpret the 256<sup>3</sup> runs (#1–#3), which represents a strong scaling analysis (in general, in image registration, strong scaling is what we’re most interested in). From 32 tasks to 512 tasks the parallel efficiency is 67%, whereas from 32 to 1024, the efficiency is 50%. This is not ideal—however, it is quite good. The majority of the calculation for low task counts goes to the interpolation computation, whereas, as we increase the number of tasks, the majority of time goes to the FFT communication phase.<sup>8</sup> Similar conclusions can be drawn for the 128<sup>3</sup> set; again, going from 16 tasks to 256 tasks, we observe 50% efficiency. For the 512<sup>3</sup> (#11–#13) the efficiency is 72%. The latter is a problem with 1.5 billion unknowns for the velocity, without counting the unknowns for the state and adjoint fields; it only takes 32 seconds to solve to an accuracy of practical interest.

If we look the weak scaling results, we can consider runs #3, #8, and #13, in which we increase the problem size by a factor of eight and the number of tasks also by a factor of

<sup>7</sup>Many image registration codes don’t even compute gradients and the termination criterion is the number of iterations. Given the limitations in the resolution and the image quality a relative reduction of the gradient by 1% is typically considered quite excessive.

<sup>8</sup>We use FFTs for the discretization of differential operators since this allows us to invert them at the cost of a spectral diagonal scaling. This offers the opportunity to exactly fulfill and *efficiently* eliminate the incompressibility constraint from the optimality system. Also, it allows for an efficient preconditioning of the Hessian with essentially no construction cost (see §III for details).





**Fig. 6:** 3D visualization of the registration problem for the brain images. From left to right: (i) reference image  $\rho_R$ , (ii) template image  $\rho_T$ , (iii) the residual differences between  $\rho_R$  and  $\rho_T$  (before registration), and (iv) the residual differences between  $\rho_R$  and  $\rho_T(\mathbf{y}_1)$  (deformed template image; after registration). Dark areas indicate a large residual and white areas no residual differences.

**Table I:** Computational performance of our solver for the synthetic registration problem illustrated in Figure 5 on TACC’s “Maverick” computing system. We neglect the incompressibility constraint for these runs. We report the time to solution, and the communication and execution times for the FFT and the interpolation, respectively (in seconds). We report timings as a function of the number of unknowns (in space), and the number of nodes and tasks. We use 16 tasks per node.

	$N^3$	nodes	tasks	FFT			interpolation	
				time to solution	communication	execution	communication	execution
#1	$64^3$	1	16	1.54	$1.20\text{E}-1$	$9.69\text{E}-2$	$1.82\text{E}-1$	$8.20\text{E}-1$
#2		2	32	$9.50\text{E}-1$	$1.42\text{E}-1$	$4.88\text{E}-2$	$1.15\text{E}-1$	$4.27\text{E}-1$
#3	$128^3$	1	16	$1.52\text{E}+1$	1.73	1.35	1.84	6.66
#4		2	32	7.88	1.30	$5.47\text{E}-1$	1.17	3.49
#5		4	64	4.70	1.19	$2.83\text{E}-1$	$5.43\text{E}-1$	1.87
#6		16	256	2.01	$6.68\text{E}-1$	$6.60\text{E}-2$	$1.86\text{E}-1$	$4.91\text{E}-1$
#7	$256^3$	2	32	$7.99\text{E}+1$	$1.44\text{E}+1$	$1.01\text{E}+1$	$1.08\text{E}+1$	$2.83\text{E}+1$
#8		8	128	$2.30\text{E}+1$	7.27	1.56	2.60	8.04
#9		32	512	7.23	2.67	$3.38\text{E}-1$	$5.93\text{E}-1$	2.00
#10		64	1024	4.72	1.70	$1.72\text{E}-1$	$4.80\text{E}-1$	1.04
#11	$512^3$	8	128	$1.91\text{E}+2$	$4.50\text{E}+1$	$2.38\text{E}+1$	$2.18\text{E}+1$	$6.89\text{E}+1$
#12		32	512	$6.07\text{E}+1$	$1.90\text{E}+1$	4.18	4.22	$1.74\text{E}+1$
#13		64	1024	$3.29\text{E}+1$	$1.28\text{E}+1$	1.77	2.33	8.57

eight. The overall timings are 15.2 seconds, 23 seconds, and 32 seconds, respectively, which again is not perfect. If we look more closely at how the time is allocated, we observe that the execution time for the FFT scales perfectly in these three runs (1.35 seconds, 1.56 seconds, and 1.77 seconds, respectively). The interpolation execution also scales well, both in terms of communication and computation. The deterioration of the overall time is due to the FFT communication costs. The largest problem we solved for the synthetic case was run #19, in which we have 3.2 billion unknowns for the velocity field on 2048 MPI tasks on “Stampede”. It took 85 seconds. The good scalability of the computation phase confirms the algorithmic optimality of the preconditioned Newton–Krylov method. We report results for the incompressible case in Table III.

### C. Real-world registration problem

We report exemplary results for the brain data sets illustrated in Figure 6 (grid size:  $256 \times 300 \times 256$ ). We set the  $\mathbf{g}_{\text{tol}}$  to  $1\text{E}-2$  and the maximal number of outer iterations (Newton steps) to 50, and  $\beta = 1\text{E}-4$ . We study strong scaling and the sensitivity of the convergence of our solver with respect to changes in the regularization weight  $\beta$ . We report scalability results for the brain images in Table IV. We display exemplary result for the

**Table V:** Sensitivity of the computational work load with respect to varying regularization weights  $\beta \in \{1\text{E}-2, 1\text{E}-3, 1\text{E}-4\}$ . We report results for four Newton iterations for the brain images. We report the number of Hessian matvecs and the time to solution (in seconds) and in parenthesis its relative increase from the base case.

	$\beta$	matvecs	time to solution ( relative )
#30	$1\text{E}-1$	43	$2.42\text{E}+1$ ( 1.0 )
#31	$1\text{E}-3$	217	$1.11\text{E}+2$ ( 4.6 )
#32	$1\text{E}-5$	1689	$8.58\text{E}+2$ ( 35.0 )

considered datasets in Figure 7. We report results for varying choices of the regularization parameter  $\beta$  in Table V.

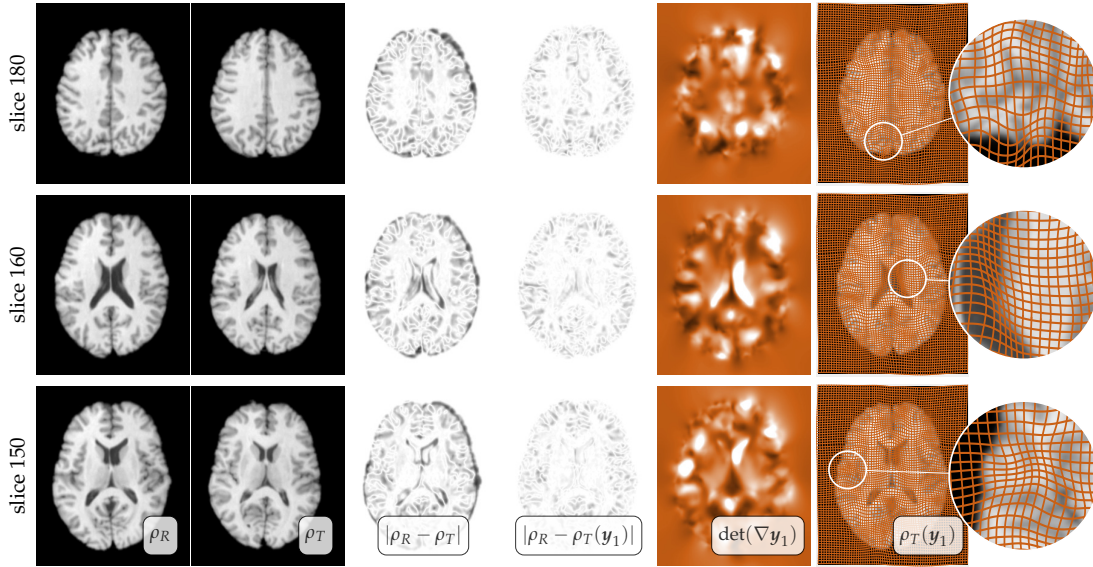
We observe that we can significantly reduce the computational timings if we switch to parallel architectures. The scaling results are consistent with what we observed for the synthetic data sets. We can reduce the wall clock time by two orders of magnitude if we change from one task on one node to 64 MPI tasks on 32 nodes. We can fit the entire problem on one node. This demonstrates the practicability of our solver. The communication and execution times of the FFT and the interpolator drop significantly as we increase

**Table II:** Computational performance of our solver for the synthetic registration problem illustrated in Figure 5 on TACC’s “Stampede” computing system. We neglect the incompressibility constraint for these runs. We report the time to solution, and the communication and execution times for the FFT and the interpolation, respectively (in seconds). We report timings as a function of the number of unknowns (in space), and the number of nodes and tasks. We use 2 tasks per node.

				FFT			interpolation	
	$N^3$	nodes	tasks	time to solution	communication	execution	communication	execution
#14	$512^3$	256	512	3.84E+1	4.61	2.62	4.12	1.98E+1
#15		512	1024	2.02E+1	2.23	1.30	2.38	9.42
#16		1024	2048	1.31E+1	1.69	6.29E−1	1.25	4.83
#17	$1024^3$	256	512	3.54E+2	3.29E+1	3.10E+1	3.72E+1	1.93E+2
#18		512	1024	1.69E+2	2.23E+1	1.39E+1	1.79E+1	8.85E+1
#19		1024	2048	8.57E+1	1.15E+1	6.75	8.78	4.42E+1

**Table III:** Computational performance of our solver for a synthetic registration problem similar to the one illustrated in Figure 5 on TACC’s “Maverick” computing system. We use the incompressibility constraint for these runs (mass preserving diffeomorphism). We report the time to solution, and the communication and execution times for the FFT and the interpolation, respectively (in seconds). We report results for a fixed grid size ( $128^3$ ) as a function of the number of nodes and tasks. We use 2 tasks per node.

			FFT			interpolation	
	nodes	tasks	time to solution	communication	execution	communication	execution
#20	1	1	1.48E+2	0	1.98E+1	2.82	9.26E+1
#21	2	4	4.27E+1	3.18	5.73	8.39E−1	2.31E+1
#22	4	8	2.25E+1	2.17	2.72	5.83E−1	1.15E+1
#23	8	16	1.09E+1	1.10	1.25	4.03E−1	5.80
#24	16	32	5.69	6.69E−1	6.20E−1	2.68E−1	2.93



**Fig. 7:** Exemplary registration results for the brain data sets. We display, from left to right axial slices of (i) the reference image  $\rho_R$ , (ii) the template image  $\rho_T$ , the residual differences (iii) between  $\rho_R$  and  $\rho_T$  before registration and (iv) after registration, (v) a point-wise map of the determinant of the deformation gradient (the color map represents volume change ranging from 0 to 2, where  $\det(\nabla \mathbf{y}_1) = 0$  is black,  $\det(\nabla \mathbf{y}_1) \in (0, 2)$  corresponds to different shades of orange (from dark to bright), and  $\det(\nabla \mathbf{y}_1) \geq 2$  is white), and (vi) the deformed template image  $\rho_T(\mathbf{y}_1)$  with a grid in overlay (closeup to the right). The values for the determinant of the deformation gradient are strictly positive (i.e., the deformation map is diffeomorphic).

**Table IV:** Strong scaling results for the brain images computed on “Maverick”. We set the regularization parameter to  $\beta = 1\text{E}-2$ . We perform two Newton iterations for these scalability runs. We report the number of nodes, the number of MPI tasks, and the communication and execution times for the FFT and the interpolation (in seconds).

	nodes	tasks	FFT			interpolation	
			time to solution	communication	execution	communication	execution
#25	1	1	1.34E+3	0.00E+1	2.59E+2	2.70E+1	7.72E+2
#26	2	4	3.92E+2	2.76E+1	6.91E+1	5.73	1.90E+2
#27	8	16	9.54E+1	8.59	1.38E+1	1.20	4.78E+1
#28	16	32	4.85E+1	4.94	6.50	5.35E-1	2.36E+1
#29	32	256	1.20E+1	4.03	1.10	8.77E-2	3.31

the number of nodes. The interpolation time contributes again critically (about or more than 50% of the time to solution).

As for the sensitivity with respect to the regularization parameter we can see that the number of Hessian matvecs (a proxy for the overall Newton-Krylov iterations) increases, as we reduce the regularization parameter from  $\beta = 1\text{E}-3$  to  $\beta = 1\text{E}-5$ . The time to solution increases by a factor of 35 for the smallest  $\beta$  reported here. This clearly demonstrates that the performance of our preconditioner is not ideal; it deteriorates with a reduction in  $\beta$ . As we have seen in the former section, the solver behaves independent of the mesh size. Implementing an improved scheme for preconditioning the Hessian requires more work.

## V. CONCLUSION

We presented a complete algorithm for large deformation diffeomorphic medical image registration. We were able to solve problems of unprecedented scale. One may ask how such runs translate to a clinical setting. As the cost of computing drops, we are hopeful that 32- and 256-task calculations will be possible at a modest cost.

The proposed algorithm is flexible and scalable. It supports different types of regularization functionals and can be extended to different image distance measures. Our approach can be easily extended to vector images and—with some additional work—can also be extended to non-stationary (time-varying) velocities [30], [9]. This will be necessary to register time-series of images or optical flow problems. All the parallelism related issues remain the same. A major remaining challenge is the design of preconditioners that are insensitive to the regularization parameter.

Finally, our algorithm relates to other applications besides medical imaging. For example applications in weather prediction and ocean physics (for tracking Lagrangian tracers in the oceans) [37], for reconstruction of porous media flows [20], and registration of Micro-CTs for material science and biology [32]. Although our method is highly optimized for regular grids with periodic boundary conditions, many aspects of our algorithm carry over.

## REFERENCES

- [1] S. S. Adavani and G. Biros, “Fast algorithms for source identification problems with elliptic PDE constraints,” *SIAM Journal on Imaging Sciences*, vol. 3, no. 4, pp. 791–808, 2008. 3
- [2] V. Akcelik, G. Biros, and O. Ghattas, “Parallel multiscale Gauss-Newton-Krylov methods for inverse wave propagation,” in *Proc ACM/IEEE Conference on Supercomputing*, 2002, pp. 1–15. 5
- [3] Y. Amit, “A nonlinear variational problem for image matching,” *SIAM Journal on Scientific Computing*, vol. 15, no. 1, pp. 207–224, 1994. 2
- [4] J. Ashburner, “A fast diffeomorphic image registration algorithm,” *NeuroImage*, vol. 38, no. 1, pp. 95–113, 2007. 2
- [5] J. Ashburner and K. J. Friston, “Diffeomorphic registration using geodesic shooting and Gauss-Newton optimisation,” *NeuroImage*, vol. 55, no. 3, pp. 954–967, 2011. 2
- [6] B. B. Avants, N. J. Tustison, G. Song, P. A. Cook *et al.*, “A reproducible evaluation of ANTs similarity metric performance in brain image registration,” *NeuroImage*, vol. 54, pp. 2033–2044, 2011. 2, 4
- [7] S. Balay, S. Abhyankar, M. F. Adams, J. Brown *et al.*, “PETSc Web page.” [Online]. Available: <http://www.mcs.anl.gov/petsc> 5
- [8] —, “PETSc users manual,” Argonne National Laboratory, Tech. Rep. ANL-95/11 - Revision 3.7, 2016. [Online]. Available: <http://www.mcs.anl.gov/petsc> 5
- [9] M. F. Beg, M. I. Miller, A. Trounev, and L. Younes, “Computing large deformation metric mappings via geodesic flows of diffeomorphisms,” *International Journal of Computer Vision*, vol. 61, no. 2, pp. 139–157, 2005. 2, 3, 4, 11
- [10] A. Borzi and V. Schulz, *Computational optimization of systems governed by partial differential equations*. Philadelphia, Pennsylvania, US: SIAM, 2012. 1, 3, 4
- [11] J. P. Boyd, *Chebyshev and Fourier spectral methods*. Mineola, New York, US: Dover, 2000. 6
- [12] M. Burger, J. Modersitzki, and L. Ruthotto, “A hyperelastic regularization energy for image registration,” *SIAM Journal on Scientific Computing*, vol. 35, no. 1, pp. B132–B148, 2013. 2
- [13] K. Chen and D. A. Lorenz, “Image sequence interpolation using optimal control,” *Journal of Mathematical Imaging and Vision*, vol. 41, pp. 222–238, 2011. 3, 4
- [14] G. E. Christensen, X. Geng, J. G. Kuhl, J. Bruss *et al.*, “Introduction to the non-rigid image registration evaluation project,” in *Proc Biomedical Image Registration*, vol. LNCS 4057, 2006, pp. 128–135. 8
- [15] G. Crippa, “The flow associated to weakly differentiable vector fields,” Ph.D. dissertation, University of Zurich, 2007. 4
- [16] K. Czechowski, C. Battaglini, C. McClanahan, K. Iyer *et al.*, “On the communication complexity of 3D FFTs and its implications for exascale,” in *Proc ACM/IEEE Conference on Supercomputing*, 2012, pp. 205–214. 6
- [17] S. C. Eisentat and H. F. Walker, “Choosing the forcing terms in an inexact Newton method,” *SIAM Journal on Scientific Computing*, vol. 17, no. 1, pp. 16–32, 1996. 5
- [18] A. Eklund, P. Dufort, D. Forsberg, and S. M. LaConte, “Medical image processing on the GPU—past, present and future,” *Medical Image Analysis*, vol. 17, no. 8, pp. 1073–1094, 2013. 2
- [19] M. Falcone and R. Ferretti, “Convergence analysis for a class of high-order semi-Lagrangian advection schemes,” *SIAM Journal on Numerical Analysis*, vol. 35, no. 3, pp. 909–940, 1998. 5
- [20] J. Fohring, E. Haber, and L. Ruthotto, “Geophysical imaging of fluid flow in porous media,” *SIAM Journal on Scientific Computing*, vol. 36, no. 5, pp. S218–S236, 2014. 11
- [21] M. Frigo and S. G. Johnson, “FFTW home page.” [Online]. Available: <http://www.fftw.org> 6
- [22] A. Gholami, J. Hill, D. Malhotra, and G. Biros, “AccFFT: A library for distributed-memory FFT on CPU and GPU architectures,” *arXiv e-*



- prints, 2016, in review (arXiv preprint: <http://arxiv.org/abs/1506.07933>). 6
- [23] A. Gholami, A. Mang, and G. Biros, “An inverse problem formulation for parameter estimation of a reaction-diffusion model of low grade gliomas,” *Journal of Mathematical Biology*, vol. 72, no. 1, pp. 409–433, 2016. 6
- [24] A. Gholami and G. Biros, “AccFFT home page.” [Online]. Available: <http://www.accfft.org> 6
- [25] A. Grama, A. Gupta, G. Karypis, and V. Kumar, *An Introduction to parallel computing: Design and analysis of algorithms*, 2nd ed. Addison Wesley, 2003. 6
- [26] M. D. Gunzburger, *Perspectives in flow control and optimization*. Philadelphia, Pennsylvania, US: SIAM, 2003. 1
- [27] M. E. Gurtin, *An introduction to continuum mechanics*, ser. Mathematics in Science and Engineering. Academic Press, 1981, vol. 158. 4
- [28] L. Ha, J. Krüger, S. Joshi, and T. C. Silva, “Multi-scale unbiased diffeomorphic atlas construction on multi-GPUs,” *GPU Computing Gems Emerald Edition*, vol. 1, pp. 771–791, 2010. 3
- [29] E. Haber and J. Modersitzki, “Numerical methods for volume preserving image registration,” *Inverse Problems*, vol. 20, pp. 1621–1638, 2004. 2
- [30] G. L. Hart, C. Zach, and M. Niethammer, “An optimal control approach for deformable registration,” in *Proc IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 9–16. 4, 11
- [31] M. Hinze, R. Pinnau, M. Ulbrich, and S. Ulbrich, *Optimization with PDE constraints*. Berlin, DE: Springer, 2009. 1
- [32] S. T. Ho and W. D. Huttmacher, “A comparison of micro CT with other techniques used in the characterization of scaffolds,” *Biomaterials*, vol. 27, no. 8, pp. 1362–1376, 2006. 11
- [33] F. Ino, K. Ooyama, and K. Hagihara, “A data distributed parallel algorithm for nonrigid image registration,” *Parallel Computing*, vol. 31, no. 1, pp. 19–43, 2005. 3
- [34] K. Ito and K. Kunisch, *Lagrange multiplier approach to variational problems and applications*. Philadelphia, Pennsylvania, US: SIAM, 2008, vol. 15. 4
- [35] G. S. James Shackelford, Nagarajan Kandasamy, *High performance deformable image registration algorithms for manycore processors*. Morgan Kaufmann, 2013. 2
- [36] R. Kakinuma, N. Moriyama, Y. Muramatsu, S. Gomi *et al.*, “Ultra-high-resolution computed tomography of the lung: Image quality of a prototype scanner,” *PloS one*, vol. 10, no. 9, p. e0137165, 2015. 1
- [37] E. Kalany, *Atmospheric modeling, data assimilation and predictability*. Oxford University Press, 2002. 11
- [38] A. Klein, J. Andersson, B. A. Ardekani, J. Ashburner *et al.*, “Evaluation of 14 nonlinear deformation algorithms applied to human brain MRI registration,” *NeuroImage*, vol. 46, no. 3, pp. 786–802, 2009. 1, 2
- [39] S. Klein, M. Staring, K. Murphy, M. A. Viergever, and J. P. W. Pluim, “ELASTIX: A toolbox for intensity-based medical image registration,” *Medical Imaging, IEEE Transactions on*, vol. 29, no. 1, pp. 196–205, 2010. 2
- [40] R. J. LeVeque, *Numerical methods for conservation laws*. Springer, 1992, vol. 132. 5
- [41] J.-L. Lions, *Some aspects of the optimal control of distributed parameter systems*. Philadelphia, Pennsylvania, US: SIAM, 1972. 1
- [42] Y. Liu, A. Fedorov, R. Kikinis, and N. Chrisochoides, “Real-time non-rigid registration of medical images on a cooperative parallel architecture,” in *Proc IEEE International Conference on Bioinformatics and Biomedicine*, 2009, pp. 401–404. 3
- [43] M. Lorenzi, N. Ayache, G. B. Frisoni, and X. Pennec, “LCC-demons: a robust and accurate symmetric diffeomorphic registration algorithm,” *NeuroImage*, vol. 81, pp. 470–483, 2013. 2
- [44] F. Lüsebrink, A. Wollrab, and O. Speck, “Cortical thickness determination of the human brain using high resolution 3T and 7T MRI data,” *Neuroimage*, vol. 70, pp. 122–131, 2013. 1
- [45] A. Mang and G. Biros, “An inexact Newton–Krylov algorithm for constrained diffeomorphic image registration,” *SIAM Journal on Imaging Sciences*, vol. 8, no. 2, pp. 1030–1069, 2015. 2, 4, 5, 6
- [46] —, “Constrained  $H^1$ -regularization schemes for diffeomorphic image registration,” *SIAM Journal on Imaging Sciences*, 2016, to appear (arXiv preprint: <http://arxiv.org/abs/1503.00757>). 2, 6
- [47] —, “A Semi-Lagrangian two-level preconditioned Newton–Krylov solver for constrained diffeomorphic image registration,” *arXiv e-prints*, 2016, in review (arXiv preprint: <http://arxiv.org/abs/1604.02153>). 2, 3, 5, 6
- [48] M. Modat, G. R. Ridgway, Z. A. Taylor, M. Lehmann *et al.*, “Fast free-form deformation using graphics processing units,” *Computer Methods and Programs in Biomedicine*, vol. 98, no. 3, pp. 278–284, 2010. 3
- [49] J. Modersitzki, *Numerical methods for image registration*. New York: Oxford University Press, 2004. 1, 2, 3
- [50] —, *FAIR: Flexible algorithms for image registration*. Philadelphia, Pennsylvania, US: SIAM, 2009. 1, 2, 3
- [51] J. Nocedal and S. J. Wright, *Numerical Optimization*. New York, New York, US: Springer, 2006. 4
- [52] J. A. Shackelford, N. Kandasamy, and G. C. Sharp, “On developing B-spline registration algorithms for multi-core processors,” *Physics in Medicine and Biology*, vol. 55, no. 21, pp. 6329–6351, 2010. 3
- [53] R. Shams, P. Sadeghi, R. A. Kennedy, and R. I. Hartley, “A survey of medical image registration on multicore and the GPU,” *Signal Processing Magazine, IEEE*, vol. 27, no. 2, pp. 50–60, 2010. 2
- [54] D. Shen and C. Davatzikos, “Very high-resolution morphometry using mass-preserving deformations and HAMMER elastic registration,” *NeuroImage*, vol. 18, no. 1, pp. 28–41, 2003. 2
- [55] A. Sotiras, C. Davatzikos, and N. Paragios, “Deformable medical image registration: A survey,” *Medical Imaging, IEEE Transactions on*, vol. 32, no. 7, pp. 1153–1190, 2013. 1, 2, 3
- [56] Z. Starosolski, C. A. Villamizar, D. Rendon, M. J. Paldino *et al.*, “Ultra high-resolution in vivo computed tomography imaging of mouse cerebrovasculature using a long circulating blood pool contrast agent,” *Scientific Reports*, vol. 5, no. 10178, 2015. 2
- [57] R. Temam, *Navier–Stokes equations: Theory and numerical analysis*. North-Holland Pub. Co., 1977. 4
- [58] R. Tomer, L. Ye, B. Hsueh, and K. Deisseroth, “Advanced CLARITY for rapid and high-resolution imaging of intact tissues,” *Nature protocols*, vol. 9, no. 7, pp. 1682–1697, 2014. 2
- [59] T. ur Rehman, E. Haber, G. Pryor, J. Melonakos, and A. Tannenbaum, “3d nonrigid registration via optimal mass transport on the GPU,” *Medical Image Analysis*, vol. 13, no. 6, pp. 931–940, 2009. 3
- [60] T. Vercauteren, X. Pennec, A. Perchant, and N. Ayache, “Symmetric log-domain diffeomorphic registration: A demons-based approach,” in *Proc Medical Image Computing and Computer-Assisted Intervention*, vol. LNCS 5241, no. 5241, 2008, pp. 754–761. 2
- [61] —, “Diffeomorphic demons: Efficient non-parametric image registration,” *NeuroImage*, vol. 45, no. 1, pp. S61–S72, 2009. 2
- [62] S. K. Warfield, M. Ferrant, X. Gallez, A. Nabavi *et al.*, “Real-time biomechanical simulation of volumetric brain deformation for image guided neurosurgery,” in *Proc ACM/IEEE Conference on Supercomputing*, 2000, pp. 23–23. 3
- [63] Y. Yin, E. A. Hoffman, and C.-L. Lin, “Mass preserving nonrigid registration of CT lung images using cubic B-spline,” *Medical Physics*, vol. 36, no. 9, pp. 4213–4222, 2009. 2
- [64] M.-Q. Zhang, L. Zhou, Q.-F. Deng, Y.-Y. Xie *et al.*, “Ultra-high-resolution 3D digitalized imaging of the cerebral angioarchitecture in rats using synchrotron radiation,” *Scientific Reports*, vol. 5, no. 14982, 2015. 2